AD-A213 661

DTIC ACCESSION NUMBER

PHOTOGRAPH THIS SHEET

LEVEL

INVENTORY 1

COMPUTER ASSISTED ADA VALIDATION TOOLS (C-A-V)

DOCUMENT IDENTIFICATION July 1988

DAJA45-87-C-0002

R/D 5677-CC-02

[illegible]

DISTRIBUTION STATEMENT

ACCESSION FOR		
NTIS	GRA&I	☒
DTIC	TAB	☐
UNANNOUNCED		☐
JUSTIFICATION	per ltr	
BY		
DISTRIBUTION /		
AVAILABILITY CODES		
DIST	AVAIL AND/OR SPECIAL	
A-1		

DISTRIBUTION STAMP

QUALITY INSPECTED 1

DATE ACCESSIONED

DATE RETURNED

89 10 3 101

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NO.

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

DTIC FORM 70A MAR 86

DOCUMENT PROCESSING SHEET

PREVIOUS EDITION MAY BE USED UNTIL STOCK IS EXHAUSTED.

NCC-THE NATIONAL CENTRE FOR INFORMATION TECHNOLOGY United Kingdom

COMPUTER ASSISTED ADA VALIDATION TOOLS

(C.A.V.)

COMPARATOR AND ANALYZER

User guide

Version: 0

Date: July 1988

Author: Maurice ASSOULINE

CONTENTS

The CAV tool ("Computer Assisted Ada Validation") regroups the comparator, the analyzer, and a user interface allowing an easy, interactive use of these tools.

The tool handles single tests, or sets of tests: a number of commands are provided for constructing all kinds of sets or subsets of tests.

1. FUNCTION OF THE ANALYZER-COMPARATOR TOOL

The Analyzer-Comparator tool CAV ("Computer assisted Ada Validation") offers the following services, through an interactive user interface:

- Automatic analysis of the execution result files of the executable tests (class A, C, D, E).

- Automatic comparison of test result files of a validation to the corresponding files of a reference validation. This comparison may operate on compilation result files (for B class tests), on link result files (for L class tests), on execution result files (for executable tests), or on any other kind of files associated to the tests.

The comparison process is able to ignore some differences between the files to compare: these non-significant differences may be specified in input files describing the result files of each validation (1 description file per validation).

- Test information handling: information on each test, including the results of the Analysis/Comparison, is stored in data files, and may be extracted from these files, either for a single test or for a given set of tests.

- Set construction: sets of tests may be built, using commands such as intersection, union, selection of tests according to a number of criteria, etc.

A few predefined sets are provided: full_set, analyze_set, compare_set, etc.

Most commands can operate on a set of tests, as well as a single test.

.2. TEST SUITES, VALIDATIONS, AND TEST STATUS

The tool CAV handles 2 main kinds of information structures: "test suites" and "validations". Each test suite or validation is implemented by a set of files. Each validation is associated to a test suite. Several validations may share the same test suite. For example, a validation and the corresponding pre-validation (reference validation) are always associated to the same test suite.

Each test suite has a name, which is used inside the names of the files of this test suite, to distinguish these files from the files of the other test suites. For example, the files of a test suite named "ACVC19" will all have names that enclose the string "ACVC19". Each validation has also a name, which is used in the same fashion inside file names.

One of the files of a test suite is the test data base file. This file contains a test record for each test of the test suite, with information such as the test name, test attributes, test kind (GLOBAL, SUBTEST, or SPLIT), and references to related test records (main subprogram subtest, original test of a SPLIT test, etc.).

Since the test data base may be used by several validations, it does not contain information about the results of the analysis/comparison for each test. This information is enclosed in a "test status" value associated to each test. The test status values are stored in a "status file" associated to each validation.

A test status may have the following values:

- UNKNOWN_YET: initial value of the status of some tests.

- IRRELEVANT: status of the subtests not associated to a result file (for example, subtests of class C, other than the main-subprogram subtests).

- TO_BE_CHECKED_BY_ANA (abbreviation: TO_ANALYZE): status of the tests that must be submitted to the analyzer.

- TO_BE_CHECKED_BY_COM (abbreviation TO_COMPARE): status of the tests that must be submitted to the comparator.

- TO_BE_CHECKED_BY_HAND (abbreviation: TO_HAND_CHECK): status of the tests that the tool has not been able to treat. Result files have to be checked manually.

- NONCONFORMANT: status of a test whose result file has been compared to the corresponding file of a reference validation, and has been found different from this reference file. (such a file must usually be checked by hand, to find out whether the differences detected by the comparator are significant).

- INAPPL_DO_SPECIAL_ACTION (abbreviation: INAP_DO_SPCL): status of the tests that need a special action to be done before they can be given the status INAPPLICABLE. Usually this special action consists in examining the result files (for example, to check the contents of some message).

- PASSED_DO_SPECIAL_ACTION (abbreviation: PASSED_DO_SPCL): status of the tests that need a special action to be done before they can be given the status PASSED. Usually this action consists in examining the result files (for example, to check the contents of some message).

- WITHDRAWN: status of the withdrawn tests of a test suite. A withdrawn test keeps the same status during the whole process of a validation.

- ANOMALOUS: status of a test whose result was not expected during the formal validation, given the pre-validation analysis, but which is judged allowable under the circumstances of the validation.

- INAPPLICABLE: status of the tests that have been accepted as inapplicable, either by the tool or by manual checking of the result files.

- FAILED: status of the tests that have been judged as failed, either by the tool or by manual checking of the result files.

- PASSED: status of the tests that have been accepted as passed, either by the tool or by manual checking of the result files.

The status value of a given test or subtest changes during the validation process. This value may be updated automatically by the analyzer/comparator, or it may be updated by hand. At the end of the validation process, all tests should have one of the following status values:

WITHDRAWN, ANOMALOUS, INAPPLICABLE, FAILED, PASSED, IRRELEVANT.

These are "final" status values. The other values are transitory, they always indicate that something has to be done to continue the treatment.

.3. TEST RECORDS

3.1. Test record kinds

The test data base of a test suite contains test records of 3 kinds: GLOBAL, SUBTEST and SPLIT.

. A test record of the kind GLOBAL represents a test as identified by the validation logbook, built up of one or several test files (test file = Ada source file of a test).

. A test record of the kind SUBTEST represents a subtest, identified by a particular test file which is only one of a number of test files associated with a GLOBAL test.

. A test record of the kind SPLIT represents the part of a test identified by a split file, which is a file created by the implementor to hold edited versions of test source files. A SPLIT kind record is related to its "original" test record, which represents the original version of a test file, not edited yet by the implementor.

. 3.2. Status of a GLOBAL test record

When a test has subtests, there is one GLOBAL record representing the complete test, and one SUBTEST record for each subtest, i.e. for each test file. The GLOBAL record is not associated to a test file: the status of this GLOBAL record is computed by the tool from the status values of all the associated subtests. For example, if the status of one of the subtests is set to the value FAILED by the comparator, or by the analyzer, or by hand, the tool will automatically set the status of the GLOBAL test record to FAILED.

If all the subtests of a test have the status PASSED, except one that has the status TO_BE_CHECKED_BY_COM, when the comparator checks this last subtest and sets the value of its status to PASSED, the status of the GLOBAL test record will automatically be changed from UNKNOWN_YET into PASSED.

When a test has no subtests and no split files, it is represented by a single test record of kind GLOBAL. This test record is associated to the single file of the test, and its status refers to this single test file.

.3.3. SPLIT records of a GLOBAL/SUBTEST record

When a test with no subtests is split, it is associated to several test files: one "original" test file, and one or more split files derived from the original file. In this case, one test record of kind SPLIT exists for the original file, and one for each split file. There is a record of kind GLOBAL representing the whole test, but it is not associated to any test file. Its status is derived from the status of all the SPLIT records, as in the case of a GLOBAL test having subtests. Note that the GLOBAL record and the SPLIT record of the original test file refer to the same test name. (This is the only case when two test records contain the same test name. When the tool needs to find a test record of this kind by its name, it will need to know which one of the 2 records must be found: the GLOBAL one or the SPLIT one).

When a subtest file has been split, there is one record of kind SUBTEST for the original file of the subtest, and one record of kind SPLIT for each split file.

.3.4. Test record of the main Ada subprogram

For the executable tests (classes A, C, D, E), only the particular subtest which is associated to the main Ada subprogram has an "interesting" status value. This "main subtest" is the only one that is associated to an execution result file. The analysis of this execution result file will determine the status of the main subtest, which is also the status of the GLOBAL test. If an executable test has no subtests, there is only a GLOBAL record, which is associated to the main Ada Subprogram, and to the execution result file. If an executable test does have subtests, one of its SUBTEST records is associated to the main Ada subprogram and to the execution result file, its GLOBAL record is not associated to a file.

The same remark applies to the L class tests: the link result file is attached to the main Ada subprogram, which is associated to a particular SUBTEST record, or to the GLOBAL record if there are no subtests.

In these two cases (executable tests and L class tests), the SUBTEST records, other than the main-subprogram one, have usually the status IRRELEVANT.

.3.5. Test attributes

Some test information is attached to the test records in the form of attributes, that a test may possess or not. The attribute information for a given test is known before the analysis/comparison of the result files: it is independent from the test results, unlike the test status.

The attributes of a test apply to the test as a whole, they concern all the test records of the test: the GLOBAL record, as well as the associated SUBTEST or SPLIT records, if there are any.

A test may have one or several of the following attributes:

- WITHDRAWN (abbreviation: W), for tests that have been suppressed from the validation, because they do not conform to the Ada standard.

- INAPPLICABLE (abbreviation: I), for tests using Ada features that are not supported by the implementation.

- SPLIT (abbreviation SPL), for tests that include one or several split files.

- MACRO (abbreviation: M), for tests that make use of implementation-dependent values: the source files of these tests must be processed (through "macro" editor commands, for example) to replace macro-names (names that begin with a dollar sign) by the actual values specified by the implementor.

- SPECIAL (abbreviation: SPE), for tests that need some special action to be done, or some special information to be collected by the validation team. (For example, check that a test has created or deleted some file, or check the contents of some information message emitted by the test.)

4. FILES USED BY THE TOOL

Several kinds of files are used by CAV:

4.1. Input files

The input files needed to run the program are:

- Test result files
- Test result file description (1 file per validation)
- Test suite description file ("TTINF" file, produced by the Tailoring Tool for a given implementation)
- User interface files (provided with the CAV tool)
- Terminal control sequences file
- Initialization file (contains the names of all the other files used by CAV, and generic file names for test result files, etc.).

4.1.1. Test result files

They include compilation result files, link result files, and execution result files. A result file must not contain the results of several tests.

The names of the result files are computed by the tool from generic names specified in the initialization file. This allows the implementor to distribute the result files as desired, and to take advantage of the hierarchical directory structure of the host system, when possible (see 4.1.6).

4.1.2. Test result file description

The format of the compilation and link result files may be different from a validation to another one, or from a pre-validation to the corresponding validation.

It is necessary to provide the comparator with information about *non-significant* differences: for example, when comparing 2 compilation result files, the number of lines that must be ignored at the beginning of each file.

It is also necessary to specify to the Analyzer the result messages that it should look for in the execution result files: for example, the strings that indicate that an executable test is PASSED, FAILED, etc. These character strings may vary slightly between 2 versions of the ACVC suite (procedure RESULT of the package REPORT).

This information, about every kind of result files of a given validation, must be specified in a "RESULT_FILE_DESCRIPTION" file. Such a file must be provided for each validation or pre-validation to be treated by the Analyzer or the Comparator.

A frame (model file) is provided for the RESULT FILE DESCRIPTION files. It contains comments that explain in detail how the model file should be completed.

The RESULT_FILE_DESCRIPTION files need not necessarily be in the current directory when running the tool: they may be placed in any directory, as long as their full path name conforms to the generic file name specified for these files in the initialization file (see 4.1.6).

4.1.3. Test suite description: TTINF file

The TTINF file is the output file produced by the Tailoring Tool. It is used by the CAV tool as an input file describing a test suite. It contains the list of the test names, the names of the subtests/splits associated to each test, the test attributes, and the indication of the main subprogram subtest.

The TTINF file is tailored for a particular implementation, it takes into account the inapplicable tests and the split tests of this implementation.

The TTINF file may be placed in any directory, as long as its full name conforms to the generic name defined in the initialization file (see 4.1.6).

.4.1.4. User interface files

The user interface files are parts of the program CAV, they are provided along with the program. They should not be modified by the validation team. The information that they hold could have been included in the program in the form of Ada text, but instead, it has been put in special text files, to allow an easier maintenance of the program, as well as a faster and better coding of the user interface.

These files are:

- The 3 menu files: Validation management, Operations on tests, Operations on sets.

- The command parameters file.

These files may be placed in any directory, as long as their full name conforms to the corresponding name specified in the initialization file (see 4.1.6).

4.1.5. Terminal control sequences file

This file contains the control sequences for the video terminal used when running the tool. The validation team must make a copy of the model file provided, and fill this copy with the control sequences that are recognized by the terminal.

In the version 0 of CAV, only a few sequences are used: it is not necessary to fill the sequences that are commented out. All the information needed to fill this file is given in the comments of the model file.

This file may be placed in any directory, as long as its full name conforms to the corresponding name specified in the initialization file (see 4.1.6). It is convenient to choose a file name that recalls or contains the name or identification of the corresponding terminal.

4.1.6. Initialization file

The initialization file is read by the program at the beginning of its execution. It contains the names of all the other files needed by the tool. Some of these names are regular file names, and some are "generic" file names. A generic file name is a pattern, used to generate several file names, parameterized by "sub-names" which may represent test names, validation names, test chapters, test classes, etc.

For example, the following generic file name will generate the names of all the link result files, in some Unix-like file system:

```
/acvc/results/valid_$V/$C$X/$T.lnk
```

$V, $C, $X and $T are special codes used as "formal parameters" of the generic name. They represent respectively the validation name, the test class, the test chapter and the test name. To generate the name of the link result file of the test LA5007C1M, in a validation named "SUN110", for example, $V will be replaced by "SUN110" $C will be replaced by "L", $X will be replaced by "A", and $T will be replaced by "LA5007C1M". The resulting name is:

```
/acvc/results/valid_SUN110/LA/LA5007C1M.lnk
```

Before running the tool, the validation team must make a copy of the model file provided for the initialization file, and specify in this copy the file names and the generic file names that should be used by the tool. All the information needed to fill the initialization file is given in the comments of the model file.

The initialization file defines also the names or generic names of some of the output files of the CAV tool. For example it contains the generic names of the validation files created by the tool for each validation (these names are parameterized by the validation name) and the generic names of the test suite files created by the tool for each test suite (parameterized by the test suite name).

The generic file names allow a fairly wide choice for the distribution of the files. Test result files, for example, may be placed in one single directory, or distributed in many ways between several directories, grouped by test class, by test chapter, etc. The only restriction is that the files corresponding to a given generic name must all have names with the same pattern: for the link result files, for example, it is not possible to have one directory per class for the chapter 3, and one single directory regrouping all classes of tests for the chapter 4. This follows from the fact that all the link result file names are generated from a single generic name.

When it is possible, it is convenient to specify the generic names for the validation files and test suite files (created by the tool) in such a way that each validation and each test suite is associated to a directory that contains all the files of this validation or test suite. For example, all the validation files could have generic names that start with the following directory name (in a Unix-like file system):

```
/acvc/valid_$V/cav_files/...
```

and, for the result files:

```
/acvc/valid_$V/results/...
```

The initialization file may have any name and may be in any directory: at the beginning of the program execution, the user will be asked for the file name. However, if the initialization file has the default file name "CAV.INI", and if it is in the current directory, the program will find it without needing to ask its name. (This default file name is defined in the program as a constant of the type STRING, in the package IMPLEM_INFO. If a different default file name is needed, you may change the value of this constant in the Ada source text).

.4.2. Files created by the tool

The program will create several kinds of files:

- Test suite files, that hold information on a test suite.
- Validation files, that hold information on a validation.
- Output files, containing the output of a command.

Usually, the name of the output file of a command may be specified by the user, when entering the command. But the names of the test suite files and validation files are constructed by the tool from the generic file names specified in the initialization file (see 4.1.6).

4.2.1. Test suite files

When a new test suite is created, the program uses the input file TTINF to create the following files:

- Test suite descriptor file: general information on the test suite.

- Test data base file: data base of the test records.

These 2 files describe completely the test suite.

The program also creates the files containing the definition of some useful sets of test records:

- Full_set: set containing all the test records.

- Withdrawn_set: set of the withdrawn test records.

- Analyze_set: set of the test records to be submitted to the Analyzer (Class A/C/D/E, main-subprogram records, not withdrawn).

- Compare_set: set of the test records to be submitted to the Comparator (Class B records associated to a source file, and class L main-subprogram records, not withdrawn).

- Special_set: set of the test records associated to a result file that needs some special hand checking (records having the SPECIAL attribute, either of the B class, or main-subprogram records of other classes, not withdrawn).

- Main_set: set of the main-subprogram records, not withdrawn.

- Global_set: set of the test records of the kind GLOBAL, not withdrawn.

4.2.2. Validation files

When a new validation is initialized, the program creates the following files, using the information available in the files of the associated test suite:

- validation descriptor file: general information on the validation.

- Status table file: file containing the current status of each test of the validation. This file is updated when leaving a session of the tool by the command "Quit". It may also be updated in the middle of a session, by the command "Update the status file".

- Logbook file: text file containing messages from the tool, for example messages from the comparator and the analyzer, indicating the name of the test currently treated. Messages are also output to the logbook file when the status of a test changes.

When an executable test is SPECIAL, its execution result file is copied into the logbook file by the Analyzer, because it usually contains information that must be checked by the validation team. This saves some time and some work to the validation team, since most of the needed information about test results may be found in the logbook file after the CAV session: there is no need to look for many individual test result files.

The logbook file is not overwritten when a new tool session starts after interruption of a previous session on the same validation: messages are appended at the end of the logbook file. (The procedure TEXT_IO.OPEN does not allow opening an existing text file to append text at the end of this file: this has been realized through the use of an auxiliary file, the "Old-logbook" file.)

Another validation file is sometimes created after a validation has already been initialized:

- The Last_treated_set file: this file contains the set of tests that has been used the most recently as a parameter of one of the following commands:

  . "Compare" (call to the Comparator)

  . "Analyze" (call to the Analyzer)

  . "Update test status"

This set of tests is saved into the file just before the command is executed, so that it is not lost in the case the command accidentally stops before being completed (because of a crash or some interruption). This allows an easy re-starting of the set treatment, either from the beginning or just for the not-treated part of the set.

4.2.3. Output files

- Several commands have an "Output file" parameter, allowing the result of a command to be written into a file specified by the user. The default output file is usually the screen (standard-output), but it may also be a file (defined by the user through the command "Change the default output file").

When a default output file is used for the output of a command, the output text is appended at the end of the default output file, which remains open for future commands. This allows grouping in a single file the output of several successive commands. When an output file, other than the default one, is used for the output of a command, this file is opened just for this command, and closed after the command is completed: it contains the output of a single command.

- The command "Write a set to a file" creates a file containing the list of the tests of a set. This file is not a text file, it contains index values refering to the test records of the test data base.

5. COMMANDS

The commands of CAV are interactive. They are put in 3 command menus:

- Validation management.
- Operations on tests.
- Operations on sets of tests.

When running the tool, the first menu that is displayed is the Validation Management menu.

From each menu, it is possible to access the other menus by typing the corresponding command. However, access to the "Operations on tests" menu or to the "Operations on sets" menu will be denied if no current validation has been previously defined (by the command "Set a validation as the current validation").

After a command has been typed, the user will be prompted for the parameters of this command, if there are any. Whenever necessary, information text will be displayed, to help the user answer the questions.

Most parameters have a default value, which is displayed on the screen together with the question asking the user for the parameter value. This default value may be selected by simply pressing the <Return> key instead of typing a value.

When a parameter has no default value, the user must type an answer, other the <Return> key: any number of carriage-return characters will be skipped when expecting a value from the keyboard.

It is always possible to cancel a command by typing "quit" instead of a parameter value, or instead of the answer to some question: the effect of the "quit" answer is always to get back to the previous interactive level: either the prompt asking for a parameter value, or the menu, or the end of the program execution (if "quit" is typed instead of a command, the execution of the tool terminates).

Every command is associated to a number in the menu, and to call a command, the user must type the associated number. However, some commands may also be called by typing an abbreviation: this is convenient for the commands that are present in several menus, with a different number in each menu.

The commands that may be called by typing an abbreviation are:

-	"Operations on tests" menu	: abbreviation "t"
-	Operations on sets" menu	: abbreviation "s"
-	"Validation management" menu	: abbreviation "v"
-	"Quit"	: may be called by typing "quit"

The user interface of CAV is not case sensitive: the answers to all questions and the parameter values may be typed in uppercase or lowercase, or both.

However, the file names typed by the user (output files for the command output) will be used with the same lettercase as specified by the user, in order to fit the file systems with case sensitive file names.

.5.1. "Validation management" commands

The validation management menu looks like the following list of commands:

	1	Initialize a test suite
	2	Initialize a validation
	3	Set a validation as the current validation
	4	Set a validation as the reference validation
	5	Change the current "Default output file"
	6	Display validation information
	7	Update the status file of a validation
	8	Close a validation
(t)	9	Operations on tests
(s)	10	Operations on sets of tests
(quit)	11	Quit

Here follows a description of each command and its parameters.

5.1.1. Initialize a test suite

Parameters:

- Name of the test suite

NOTE: The test suite name that you will type will be inserted in character strings (defined in the initialization file) to generate the names of the test suite files. If the file names generated this way include directory names, these directories must exist, otherwise the initialization of the test suit will fail.

Action:

This command constructs the name of the TTINF file of the test suite, opens this file, and extracts data from this file to create the test suite files. (Mainly the test_data_base file.)

5.1.2. Initialize a validation

Parameters:

- Name of the validation

NOTE: The validation name that you will type will be inserted in character strings (defined in the initialization file) to generate the names of the validation files. If the file names generated this way include directory names, these directories must exist, otherwise the initialization of the validation will fail.

- Is it a pre-validation ? (Yes/No)

- Name of the associated test suite

Action:

This command creates the files of the specified validation, except if they already exist ; In particular, it initializes the contents of the status file. Then, it sets the specified validation as the current validation (see 5.1.3).

5.1.3. Set a validation as the current validation

Parameters:

- Name of the validation

Action:

This command opens the files of the specified validation and loads information from these files, except if it has been previously done. Then, it sets this validation as the current one, i.e. the default one used for operations on tests or sets of tests.

If this validation has been used previously in the same session, the sets of tests that have been constructed previously become accessible again, and the previous values of the "current set" and "current test" are restored.

5.1.4. Set a validation as the reference validation

Parameters:

- Name of the validation

Action:

This command opens the files of the specified validation and loads information from these files, except if it has been previously done. Then, it sets this validation as the reference validation, which is the validation that provides the test result files to be compared to the corresponding files of the current validation.

NOTE: the reference validation and the current validation must be associated to the same test suite, in order to ensure that each test file of the current validation matches an existing reference file.

5.1.5. Change the current "Default output file"

Parameters:

- Name of the new default output file
  Default: STANDARD_OUTPUT

Action:

If the specified file name denotes an existing file, the user is asked to confirm whether this file may be overwritten. If the answer is "yes", or if the file name does not denote an existing file, this command closes the previous "default output file", unless it was STANDARD_OUTPUT, and opens the specified file, which becomes the "default output file".

The "default output file" is used for the output of some commands, when the user has not specified a particular output file for the command.

The output of several commands may be sent to the default output file: each command output will be appended at the end of the file, and will not overwrite the previous contents of the file (as long as the default output file does not change). On the contrary, when a particular output file is specified for a command, it remains open only during this command execution, and it contains only the output of this single command.

Initially, the default output file is STANDARD_OUTPUT.

.5.1.6. Display validation information

Parameters:

- Name of the validation
  Default: current validation, if there is one (see 5.1.3)

- Output file name
  Default: current "Default output file" (see 5.1.5)

NOTE: even if the default output file is not the STANDARD_OUTPUT, you may obtain a display on the STANDARD_OUTPUT by entering the character '>' instead of a file name.

Action:

If an output file has been specified and if it exists, the user is asked to confirm whether this file may be overwritten. Then, this command opens the files of the specified validation and loads information from these files, except if it has been previously done.

Then, general information on the specified validation is printed to the output file. This information includes the name of the associated test suite, the number of global tests, test counts by attribute, whether the specified validation is the current one, the reference one, etc.

If the output file is not the default one, it is left closed.

If the current validation was not previously defined, the specified validation becomes the current one.

.5.1.7. Update the status file of a validation

Parameters·

- Name of the validation
  Default: current validation, if there is one (see 5.1.3.)

Action:

If no status value has been modified since the last update of the status file, no action takes place. Otherwise, this command writes to the status file of the specified validation the current status values of the test records.

Since the status values of the specified validation must be present in memory, one of the following commands must have been previously called for the specified validation:

Initialize a validation, set a validation as the current validation, set a validation as the reference validation, display validation information.

.5.1.8. Close a validation

Parameters:

- Name of the validation
  Default: current validation, if there is one (see 5.1.3)

- Update status file ? (Yes/No)
  Default: Yes

Action:

This command deallocates the information about the specified validation that has been previously loaded in memory, and closes any open file associated to the specified validation. This command is useful when information has been loaded concerning too many validations, preventing the user from "opening" another validation.

If the status values of the tests records have been modified since the last update of the status file, and if the updating of the status file has been requested (2nd parameter), the current status values are written to the status file, as in the command 5.1.7.

The specified validation must be "open": in other words, information about this validation must have been previously loaded through one of the following commands: Initialize a validation, Set a validation as the current validation, Set a validation as the reference validation, Display validation information.

If some sets of tests have been constructed for the specified validation, and have not been saved into files (see 5.3.3), the user will be asked to confirm that these sets may be discarded (and lost). If the user does not confirm, the validation will not be closed, so that the user may save the sets into files.

Before calling the command "Quit" (see 5.1.11), all the open validations, except the current validation, must be closed: this is to ensure that the data created or updated during the current session (status values, sets of tests) will not be lost accidentally when terminating the session.

5.1.9. Operations on tests

Parameters:

- None

Action:

This command calls another menu, giving access to all the operations on tests of the current validation: analyze result files, compare, etc. (see 5.2).

The current validation must have been previously defined (see 5.1.2, 5.1.3, 5.1.6).

5.1.10. Operations on sets of tests

Parameters:

- None

Action:

This command calls another menu, giving access to all the operations on sets of tests of the current validation: set construction, set information, reading a set from a set file, writing a set to a set file, etc.

The current validation must have been previously defined (see 5.1.2, 5.1.3, 5.1.6).

5.1.11. Quit

Parameters:

- Update the status file of the current validation ? (Yes/No)
  Default: Yes

Action:

If any validation, other than the current one, is open, this command is not executed: the user is asked to close all these open validations (see 5.1.8). Otherwise, the status file of the current validation is updated, if this has been requested and if some status value has changed since the last update of the status file. Then the current validation is closed (see 5.1.8), all the open files are closed, and the program execution terminates.

5.2. "Operations on tests" commands

The "Operations on tests" menu looks like the following list of commands:

```
               1  (Re)define the current test or subtest
               2  Display test information (for a single test or
                  for a set)
               3  Analyze execution result files
               4  Compare test result files (to the reference
                  validation ones)
               5  Update test status
               6  Make a progress report of the current validation
        (v)    7  Validation management operations
        (s)    8  Operations on sets of tests
        (quit) 9  Quit
```

Here follows a description of each command and its parameters:

5.2.1. (Re)define the current test or subtest

Parameters:

- Test record name

NOTE: if the name specified by the user is ambiguous (particular situation described in section 3.3), the user will be asked to specify which one of the homonym test records (GLOBAL or SPLIT record) he means.

Action:

The specified test record becomes the current default one: it will be taken as the default test record for the next operation on tests.

5.2.2. Display test information

Parameters:

- Test record name, or Set name or number, preceded by the keyword "set".
  Default: current test (see 5.2.1), or current set (see 5.3.5), if the keyword "set" is typed.

- Level of detail (Name_only, Name_and_status, Brief, or Full).
  Default: Brief for a single test, Name_only for a set.

- Output file name
  Default: current "Default output file" (see 5.1.5)

NOTE: even if the default output file is not the STANDARD_OUPUT, you may obtain a display on the STANDARD_OUTPUT by entering the character '>' instead of a file name.

Action:

This command displays information about the specified test or set of tests, on the specified output file. If the default output file is used, the output text is appended at the end of this file, and the file remains open for the output of further commands. If a particular file name is specified, it is created, it receives the output text, then it is closed. In the case the specified file name matches an existing file, the user is asked to confirm whether he wants to overwrite it, or to specify another file name.

According to the specified detail level, the following information is displayed for the specified test or set of tests:

Name_only: the test record kind and name.

Name_and_status: the test record kind, name, and status.

Brief: the test record kind, name, status, attributes, and, if relevant, the number of subtests, number of splits, name of the main subtest, name of the original split test.

NOTE: the status and the attributes are displayed in their abbreviated form, defined in sections 2 and 3.5.

Full: same information as for Brief, not only for the specified test records, but also for each subordinate test record: for a GLOBAL record, subordinate test records are all the associated SUBTEST or SPLIT records ; for a SUBTEST record, subordinate test records are the associated SPLIT records, if there are any.

When displaying information for a set of tests, in the mode "Full", redundant information is eliminated. For example, if the set contains a GLOBAL test record and one of its SUBTEST records, the SUBTEST record information will be displayed only once (instead of twice: once as a subordinate record of the GLOBAL one, and once as a record belonging to the set).

5.2.3 Analyze execution result files

Parameters:

- Test record name, or Set name or number, preceded by the keyword "set".
  Default: current test (see 5.2.1), or current set (see 5.3.5), if the keyword "set" is typed.

- Output file name
  Default: current "Default output file" (see 5.1.5).

NOTE 1: even if the default output file is not the STANDARD_OUTPUT, you may obtain a display on the STANDARD_OUTPUT by entering the character '>' instead of a file name.

NOTE 2: the same output test is also appended to the logbook file of the current validation (see 4.2.2).

Action:

This command opens and analyzes the execution result file(s) of the specified test or set of tests, then updates the test status according to the results of the analysis.

If the first parameter is a set, this set is saved in the LAST_TREATED_SET file of the current validation before the treatment (see 4.2.2). This allows an easy restarting of the same treatment from a certain point, in case of interruption.

The output of the analysis contains the new status of the treated test(s), as well as any other relevant messages. It also contains, for the SPECIAL tests, the whole text of the execution result file. This text is useful to the validation team, because it usually has to be checked for special information.

The output is sent to the specified output file, and it is also appended at the end of the logbook file of the current validation (see 4.2.2). If the default output file is used, the output text is appended at the end of this file, and the file remains open for the output of further commands. If a particular file name is specified, it is created, it receives the output text, then it is closed. In the case the specified file name matches an existing file, the user is asked to confirm whether he wants to overwrite it, or to specify another file name.

The analysis consists in opening the execution result file, looking at the end of this file for the result message, and updating the status of the test record, taking into account the contents of this message, and also the kind of the test record, its attributes, etc. In the case of a SUBTEST record (this record must represent the main subprogram subtest), the status of the associated GLOBAL test record will also be updated.

Note that the test status are updated in main memory only: the status file will be updated when one of the following commands is executed: "Quit" (5.1.11), "Update the status file of a validation" (5.1.7), or "Close a validation" (5.1.8).

To take into account the differences that may exist between the versions of the ACVC test suite, the analysis is parameterized by a description of the execution result files. This description must be provided in the RESULT_FILE_DESCRIPTION file of the current validation (see 4.1.2).

Before calling this command, the RESULT_FILE_DESCRIPTION file of the current validation must exist and be correctly filled, at least for the part corresponding to the execution result file analysis.

To fill the RESULT_FILE_DESCRIPTION file, please read the indications in the comments of the model file provided along with the program.

5.2.4. Compare test result files

Parameters:

- Test record name, or Set name or number, preceded by the keyword "set".
  Default: current test (see 5.2.1), or current set (see 5.3.5), if the keyword "set" is typed.

- Kind of test file to compare, to be selected between:

  . DEFAULT KIND corresponding to the class of the test, that is:
    . COMPILATION_RESULT for class B tests,
    . LINK_RESULT for class L tests, and
    . EXECUTION_RESULT for other classes.
  . COMPILATION RESULT
  . LINK RESULT
  . EXECUTION RESULT
  . EXTRA FILE, defined in the initialization file

- Output file name
  Default: current "Default output file" (see 5.1.5)

NOTE 1: even if the default output file is not the STANDARD-OUTPUT, you may obtain a display on the STANDARD_OUTPUT by entering the character '>' instead of a file name.

NOTE 2: the same output text is also appended to the logbook file of the current validation (see 4.2.2).

Action:

This command opens the specified test files of the current validation, and compares them to the corresponding files of the reference validation, then updates the test status, according to the results of the comparison.

If the first parameter is a set, this set is saved in the LAST_TREATED_SET file of the current validation before the treatment (see 4.2.2). This allows an easy restarting of the same treatment from a certain point, in case of interruption.

The output of the comparison contains the new status of the treated test(s), as well as any other relevant messages. If the files are different, the line numbers of the first different lines found are indicated.

The output is sent to the specified output file, and it is also appended at the end of the logbook file of the current validation (see 4.2.2). If the default output file is used, the output text is appended at the end of this file, and the file remains open for the output of further commands. If a particular file name is specified, it is created, it receives the output text, then it is closed. In the case the specified file name matches an existing file, the user is asked to confirm whether he wants to overwrite it, or to specify another file name.

The comparison consists in opening the files to compare, comparing each line of the "subject file" to the corresponding line of the "reference file", ignoring any unsignificant differences (these differences to ignore are specified by the RESULT_FILE_DESCRIPTION files), and updating the status of the test record, in the current validation, according to the result of the comparison, and also to the kind of the test record, its attributes, etc.

When the status of a test record is updated, any status change involved in other test records is also made. For example, if the compilation result files of all the subtests of a B class test have been compared successfully, and the PASSED status has been given to the corresponding SUBTEST records, then the PASSED status has also been attributed to the GLOBAL test record, at the time the last SUBTEST status was updated (see also 3.2, 3.3, 3.4).

Note that the test status are updated in main memory only: the status file will be updated when one of the following commands is executed: "Quit" (5.1.11), "Update the status file of a validation" (5.1.7), or "Close a validation" (5.1.8).

It is possible to compare any kind of files attached to the test records, by specifying "EXTRA FILE" for the 2nd parameter. The name of these "extra files" will be constructed in the same way as the other test file names, using the corresponding generic file name specified in the initialization file. Of course this generic file name must be correctly filled in the initialization file before trying to compare the "EXTRA TEST FILES".

The test status is updated only when comparing the right kind of files, according to the test class: for executable tests, for example, only the comparison of the execution result files will lead to updating the test status. For the "EXTRA TEST FILES", the status is never updated. However, a message is displayed, stating that the files are identical or different, and indicating the numbers of the first different lines.

The RESULT_FILE_DESCRIPTION files of the current validation and the reference validation are used by this command to identify the lines to ignore in the test result files. They contain, for each kind of file, information such as:

- number of lines to skip at the beginning of the file,
- character string delimiting the lines to ignore at the beginning of the file,
- number of lines to skip at the beginning of each page,
- etc. (see the comments in the model file).

Of course, each validation has its description file, wich may contain different values of the parameters.

Before calling this command:

- a reference validation must be defined (see 5.1.4.),
- the RESULT_FILE_DESCRIPTION files of the current validation and of the reference validation must exist ande be correctly filled, at least for the part corresponding to the kind of files to compare.

To fill the RESULT_FILE_DESCRIPTION files, please read all the indications in the comments of the model file provided.

5.2.5. Update test status

Parameters:

- Test record name, or Set name or number, preceded by the keyword "set".
  Default: current test (see 5.2.1.), or current set (see 5.3.5.), if the keyword "set" is typed.

- New status, to be chosen between:

  1. UNKNOWN_YET
  2. IRRELEVANT
  3. TO_BE_CHECKED_BY_ANA
  4. TO_BE_CHECKED_BY_COM
  5. TO_BE_CHECKED_BY_HAND
  6. NONCONFORMANT
  7. INAPPL_DO_SPECIAL_ACTION
  8. PASSED_DO_SPECIAL_ACTION
  9. WITHDRAWN
  10. ANOMALOUS
  11. INAPPLICABLE
  12. FAILED
  13. PASSED

  (see section 2 for the meaning of these values).

- Output file name
  Default: current "Default output file" (see 5.1.5)

NOTE 1: even if the default output file is not the STANDARD_OUTPUT, you may obtain a display on the STANDARD_OUTPUT by entering the character '>' instead of a file name.

NOTE 2: the same output text is also appended to the logbook file of the current validation (see 4.2.2).

Action

This command is used to update "by hand" the status of a test record. It sets the status of the specified test or set of tests to the specified value, and displays a log message as well as any other relevant messages.

If the first parameter is a set, this set is saved in the LAST_TREATED_SET file of the current validation before the treatment (see 4.2.2). This allows an easy restarting of the same treatment from a certain point, in case of interruption.

The output is sent to the specified output file, and it is also appended at the end of the logbook file of the current validation (see 4.2.2). If the default output file is used, the output text is appended at the end of this file, and the file remains open for the output of further commands. If a particular file name is specified, it is created, it receives the output text, then it is closed. In the case the specified file name matches an existing file, the user is asked to confirm whether he wants to overwrite it, or to specify another file name.

When a test status is updated "by hand", any status change involved in other test records is also made, just as in the case the status is updated by the "Analyze" command (5.2.3) or the "Compare" command (5.2.4). The corresponding log messages are displayed, letting the user know that some other status has been updated.

Note that the test status are updated in main memory only: the status file will be updated when one of the following commands is executed: "Quit" (5.1.11), "Update the status file of a validation" (5.1.7), or "Close a validation" (5.1.8).

5.2.6. Make a progress report of the current validation

This command is not implemented in version 0 of the program.

5.2.7 Validation management operations

Parameters:

- None

Action:

This command calls the menu giving access to the validation management operations: initialize a validation, display validation information, etc. (see 5.1).

5.2.8. Operations on sets of tests (same command as 5.1.10)

Parameters:

- None

Action:

This command calls another menu, giving access to all the operations on sets of tests of the current validation: set construction, set information, reading a set from a set file, writing a set to a set file, etc.

The current validation must have been previously defined (see 5.1.2, 5.1.3, 5.1.6).

5.2.9. Quit (same command as 5.1.11)

Parameters:

- Update the status file of the current validation ? (Yes/No)
  Default: Yes

Action:

If any validation, other than the current one, is open, this command is not executed: the user is asked to close all these open validations (see 5.1.8). Otherwise, the status file of the current validation is updated, if this has been requested and if some status value has changed since the last update of the status file. Then the current validation is closed (see 5.1.8), all the open files are closed, and the program execution terminates.

5.3. "Operations on sets" commands

The "Operations on sets" menu looks like the following list of commands:

	1	List the sets of the current validation
	2	Display test information (for a single test or for a set)
	3	Write a set to a file
	4	Read a set from a file
	5	Redefine the current set
	6	Add a test to a set
	7	Remove a test from a set
	8	Union of two sets
	9	Intersection of two sets
	10	Difference of two sets
	11	Extract a subset from a set
	12	Display the history of the sets
(v)	13	Validation management operations
(t)	14	Operations on tests
(quit)	15	Quit

A certain number of sets are kept in memory, so that they can be available for operations on tests or on sets. These sets are separated in 2 parts:

- Predefined sets of the associated test suite (these sets are described in 4.2.1)

- User-constructed sets (at the beginning of a session, these sets are empty).

Each set occupies a set entry. There are 7 set entries for the predefined sets, these entries are numbered from 1 to 7. There are 5 set entries for the user-constructed sets, numbered from 8 to 12.

When specifying a set as a command parameter, this set may be identified either by its entry number, or by its name: either form may be typed. When creating a new set as the result of a set operation, the user must choose a name for this set, and he must choose an entry number for this set. If this entry was already assigned to a set, the user will be asked to confirm whether he wants to loose (deallocate) the previous set. The 7 entries reserved to the predefined sets may not be used for a new set: they are "read-only" entries.

Here follows a description of each command and its parameters:

5.3.1. List the sets of the current validation

Parameters:

- Output file name
  Default: STANDARD_OUTPUT

Action:

This command displays a list of the sets of the current validation, which are available for any operation on tests or sets of tests. This list is separated in 2 parts:

- Predefined sets of the associated test suite (these sets are described in 4.2.1)

- User constructed sets. At the beginning of a session, these sets are all empty.

For each set, a line containing the following information is displayed:

- set entry number

- set name (18 characters or less)

- number of test records (cardinality of the set)

- status letter U, S, or N:
  U stands for "Unused"
  S stands for "Saved" (into a set file)
  N stands for "Not saved".

- Comment text (48 characters or less) about the set. This text is provided by the user for the user-constructed sets, it helps him remember the contents of the sets.

The current default set is indicated by a star character ('*') at the beginning of the line.

If an output file has been specified, it is created, it receives the output text, then it is closed. In the case the specified file name matches an existing file, the user is asked to confirm whether he wants to overwrite it, or to specify another file name.

5.3.2. Display test information (same command as 5.2.2.)

Parameters:

- Test record name, or Set name or number, preceded by the keyword "set".
  Default: current test (see 5.2.1), or current set (see 5.3.5), if the keyword "set" is typed.

- Level of detail (Name_only, Name_and_status, Brief, or Full).
  Default: Brief for a single test, Name_only for a set.

- Output file name
  Default: current "Default output file" (see 5.1.5)

NOTE: even if the default output file is not the STANDARD_OUPUT, you may obtain a display on the STANDARD_OUTPUT by entering the character '>' instead of a file name.

Action:

This command displays information about the specified test or set of tests, on the specified output file. If the default output file is used, the output text is appended at the end of this file, and the file remains open for the output of further commands. If a particular file name is specified, it is created, it receives the output text, then it is closed. In the case the specified file name matches an existing file, the user is asked to confirm whether he wants to overwrite it, or to specify another file name.

According to the specified detail level, the following information is displayed for the specified test or set of tests:

Name_only: the test record kind and name.

Name_and_status: the test record kind, name, and status.

Brief: the test record kind, name, status, attributes, and, if relevant, the number of subtests, number of splits, name of the main subtest, name of the original split test.

NOTE: the status and the attributes are displayed in their abbreviated form, defined in sections 2 and 3.5.

Full: same information as for Brief, not only for the specified test records, but also for each subordinate test record: for a GLOBAL record, subordinate test records are all the associated SUBTEST or SPLIT records ; for a SUBTEST record, subordinate test records are the associated SPLIT records, if there are any.

When displaying information for a set of tests, in the mode 'Full", redundant information is eliminated. For example, if the set contains a GLOBAL test record and one of its SUBTEST records, the SUBTEST record information will be displayed only once (instead of twice: once as a subordinate record of the GLOBAL one, and once as a record belonging to the set).

5.3.3. Write a set to a file

Parameters:

- Set identification (set entry number, or set name)
  Default: current default set.

- Name of the file

Action:

This command creates a file with the specified name, saves the specified set into this file, then closes it. In the case the specified name matches an existing file, the user is asked to confirm whether he wants to overwrite it, or to specify another file name.

5.3.4. Read a set from a file

Parameters:

- Name of the file (this file must be a "set file", created by the command 5.3.3).

- Set entry number, for the result set (number between 8 and 12).

- Name of the result set (18 characters or less)
  Default: the name of the file, truncated to 18 characters.

- Comment text associated to the result set (48 characters or less).
  Default: empty text.

Action:

This command opens the specified file, reads the set it contains, closes the file, and assigns the specified entry number to the set.

If this entry was already occupied by a set, the user is asked to confirm that he wants to deallocate this previous set (and loose it, if it has not been saved into a file).

The result set becomes the current default set.

5.3.5. Redefine the current set

Parameters:

- Set identification (set entry number, or set name).

- New name for the specified set (18 characters or less).
  Default: previous name, unchanged

- New comment text for the specified set (48 characters or less).
  Default: previous text, unchanged.

Action:

The specified set becomes the current default set. If a new name or comment has been specified, this name or comment is attached to the specified set, replacing the previous name or comment.

5.3.6. Add a test to a set

Parameters:

- Set identification (set entry number, or set name)
  Default: current default set.

- Test name
  Default: current test name

- Set entry number, for the result set.
  (It is possible to specify the entry number of the "operand" set, used as the first parameter, if this set is not needed anymore).

- Name of the result set (18 characters or less).
  Default: the previous name of the result set entry, unchanged.

- Comment text associated to the result set (48 characters, or less).
  Default: the same text previously associated to the result set entry.

Action:

This command adds the specified test to the specified set (if the set does not already contain this test), and assigns the specified entry number to the resulting set.

If this entry was already occupied by a set, the user is asked to confirm that he wants to deallocate this previous set (and loose it, if it has not been saved into a file).

The result set becomes the current default set.

5.3.7. Remove a test from a set

Parameters:

- Set identification (set entry number, or set name)
  Default: current default set.

- Test name
  Default: current test name

- Set entry number, for the result set.
  (It is possible to specify the entry number of the "operand" set, used as the first parameter, if this set is not needed anymore).

- Name of the result set (18 characters or less).
  Default: the previous name of the result set entry, unchanged.

- Comment text associated to the result set (48 characters, or less).
  Default: the same text previously associated to the result set entry.

Action:

This commands removes the specified test from the specified set (if the set contains this test), and assigns the specified entry number to the resulting set. If this entry was already occupied by a set, the user is asked to confirm that he wants to deallocate this previous set (and loose it, if it has not been saved into a file).

The result set becomes the current default set.

5.3.8. Union of two sets

Parameters:

- Set identification of the first set (set entry number, or set name).
  Default: current default set

- Set identification of the second set (set entry number, or set name).

- Set entry number, for the result set. (It is possible to specify the entry number of one of the "operand" sets, if this set is not needed anymore).

- Name of the result set (18 characters or less).
  Default: the previous name of the result set entry, unchanged.

- Comment text associated to the result set (48 characters, or less).
  Default: the same text previously associated to the result set entry.

Action:

This command constructs the union of the 2 specified sets: the set containing the test records that belong to one at least of the two operand sets.

The resulting set is attached to the specified entry number. If this entry was already occupied by a set, the user is asked to confirm that he wants to deallocate this previous set (and loose it, if it has not been saved into a file).

The result set becomes the current default set.

5.3.9. Intersection of two sets

Parameters:

- Set identification of the first set (set entry number, or set name).
  Default: current default set

- Set identification of the second set (set entry number, or set name).

- Set entry number, for the result set.
  (It is possible to specify the entry number of one of the "operand" sets, if this set is not needed anymore).

- Name of the result set (18 characters or less).
  Default: the previous name of the result set entry, unchanged.

- Comment text associated to the result set (48 characters, or less).
  Default: the same text previously associated to the result set entry.

Action:

This command constructs the intersection of the 2 specified sets: the set containing the test records that belong to both operand sets.

The resulting set is attached to the specified entry number. If this entry was already occupied by a set, the user is asked to confirm that he wants to deallocate this previous set (and loose it, if it has not been saved into a file).

The result set becomes the current default set.

5.3.10. Difference of two sets

Parameters:

- Set identification of the first set (set entry number, or set name).
  Default: current default set.

- Set identification of the second set (set entry number, on set name).

- Set entry number, for the result set.
  (It is possible to specify the entry number of one of the "operand" sets, if this set is not needed anymore).

- Name of the result set (18 characters or less).
  Default: the previous name of the result set entry, unchanged.

- Comment text associated to the result set (48 characters, or less).
  Default: the same text previously associated to the result set entry.

Action:

This command constructs the asymmetrical difference of the 2 specified sets: the set containing the test records that belong to the first set and do not belong to the second one.

The resulting set is attached to the specified entry number.

If this entry was already occupied by a set, the user is asked to confirm that he wants to deallocate this previous set (and loose it, if it has not been saved into a file).

The result set becomes the current default set.

5.3.11. Extract a subset from a set

Parameters:

- Set identification of the input set (see entry number, or set name).
  Default: current default set

- Pattern for the names of the test records to select. This pattern is a regular expression that may contain the wildcard character '*'. For example, the pattern "C5*B" will match all the test records whose name starts with "C5" and ends with "B".
  Default: *

- Test name for the lower bound of the range to select. This parameter may be used to select only the test records from a given test, in the logbook order.
  Default: first test of the logbook.

- Test name for the upper bound of the range to select. This parameter may be used to select only the test records that come before a given test, in the logbook order.
  Default: last test of the logbook.

- Test record kinds of the records to select: one or several kinds, among GLOBAL, SUBTEST, SPLIT.
  Default: any kind

- Status of the test records to select.
  Default: any status

- Attributes required from the test records to select: one or several attributes, among WITHDRAWN, INAPPLICABLE, SPLIT, MACRO, SPECIAL.
  Default: none.

- Attributes refused, causing the exclusion from the selection: one or several attributes, among WITHDRAWN, INAPPLICABLE, SPLIT, MACRO, SPECIAL.
  Default: none.

- Set entry number, for the result set.
  (It is possible to specify the entry number of the input set, used as the first parameter, if this set is not needed anymore).

- Name of the result set (18 characters or less).
  Default: the previous name of the result set entry, unchanged.

- Comment text associated to the result set (48 characters, or less).
  Default: the same text previously associated to the result set entry.

Action:

This command constructs a subset of the input set by selecting only the test records that match the specified selection criteria, and assigns the specified result set entry number to this subset. If this entry was already occupied by a set, the user is asked to confirm that he wants to deallocate this previous set (and loose it, if it has not been saved into a file).

The result set becomes the current default set.

5.3.12. Display the history of the sets

This command is not implemented in version 0 of the program.

5.3.13. Validation management operations (same command as 5.2.7)

Parameters:

- None

Action:

This command calls the menu giving access to the validation management operations: initialize a validation, display validation information, etc. (see 5.1).

5.3.14. Operations on tests (same command as 5.1.9)

Parameters:

- None

Action:

This command calls another menu, giving access to all the operations on tests of the current validation: analyze result files, compare, etc. (see 5.2).

The current validation must have been previously defined (see 5.1.2, 5.1.3, 5.1.6).

5.3.15. Quit (same command as 5.1.11)

Parameters:

- Update the status file of the current validation ? (Yes/No)
  Default: Yes

Action:

If any validation, other than the current one, is open, this command is not executed: the user is asked to close all these open validations (see 5.1.8). Otherwise, the status file of the current validation is updated, if this has been requested and if some status value has changed since the last update of the status file. Then the current validation is closed (see 5.1.8), all the open files are closed, and the program execution terminates.

6. TOOL INSTALLATION

NOTE: the file names mentionned in this section are the ones that have been used in the development version of the tool, on a MS-DOS file system. Of course, these names are just provided as an example, since they must be modified to satisfy the requirements of the file name syntax on other systems.

6.1. Compiling and linking the program

The set of files provided include the source files of the program, as well as the file COMPIL.ORD which indicates a correct compilation order for the source files.

To create an executable file:

- create an Ada library
- compile the source files, using the compilation order provided in the file COMPIL.ORD
- link the program: the main subprogram name is CAV.

6.2. Filling the initialization file (CAV.INI)

- Make a copy of the model file provided for the initialization file CAV.INI.
- Read section 4.1.6, for general information about the initialization file.
- Fill the copy of the model file with the file names that you want to use when executing the program. Read carefully the information contained in the comments of this file, about the format of the file. You may also use the example file provided.
- Preferably, put this file in the current directory when executing the program (see 4.1.6).

6.3. User interface files

The user interface files, described in section 4.1.4., are:

```
VAL_MGT.MNU
TEST_OP.MNU
SET_OP.MNU
COMMANDS.PAR
```

- Do not modify the contents of these files.
- Rename them or move them according to their name, which should be now defined in the initialization file.

6.4. Terminal control sequence file

The terminal control sequence file is described in section 4.1.5.

- Make a copy of the model file provided (TERMINAL.SEQ), and rename it or move it according to its name, that you have previously defined in the initialization file. It is convenient to choose a file name that recalls or contains the name or identification of the terminal you will be using.

- Edit this copy, replacing the control sequences it contains by the sequences recognized by your terminal. It is not necessary to fill the sequences that are commented out. Use the example file provided (IBM_PCAT.SEQ), and read carefully the indications in the comments.

6.5. TTINF and test suite files

They consist of :

- The TTINF file, produced by the Tailoring Tool, and used by the CAV tool (see 4.1.3).
- The files produced by CAV for each test suite (sec. 4.2.1).

Each kind of test suite file has a generic file name, defined in the initialization file. This generic name may include the characters "$U", which represent a test suite name. (It is convenient, when possible, to have one directory for each test suite: this may be done simply by including "$U" in the directory path of the generic names of the test suite files.)

For each test suite that you will use:

- Choose a test suite name.
- From the generic names (of test suite files) that you have defined in the initialization file, figure out the actual file names by replacing "$U" by the test suite name. If these actual file names contain directory names, create these directories if they do not exist (so that the tool can create the test suite files).
- Move or rename the TTINF file (produced by the Tailoring Tool for this test suite) according to its generic name, defined in the initialization file.

6.6. Validation files

The validation files are:

- The RESULT_FILE_DESCRIPTION file (RESULT_F.DES) (see 4.1.2).

- The files produced by CAV for each validation (see 4.2.2)

Each kind of validation file has a generic file name, defined in the initialization file. This generic name may include the characters "$V", which represent a validation name. (It is convenient, when possible, to have one directory for each validation: this may be done simply by including "$V" in the directory path of the generic names of the validation files.)

For each validation that you will use, either as the current *validation or the reference validation:*

- Choose a validation name.

- From the generic names (of validation files) that you have defined in the initialization file, figure out the actual file names by replacing "$V" by the validation name. If these actual file names contain directory names, create these directories if they do not exist (so that the tool can create the validation files).

- Make a copy of the model file provided for the RESULT_FILE_DESCRIPTION file (RESULT_F.DES), and rename it or move it according to its "actual name". Generate this actual name from the generic name defined in the initialization file, replacing "$V" by the validation name.

- Edit this copy, following the indications it contains.

6.7. Test result files

Check that the test result files have names that conform with the generic names of the initialization file.

If not, you may:

- Either change the generic names to make them represent correctly the whole set of result files,

- Or move/rename the test result files in conformance with the generic file names,

- Or do a combination of both.

7. TOOL EXECUTION

The following steps and commands give an idea of how the CAV tool may be used. They are mentionned just as an example: feel free to use the tool in a different way:

- Initialize a test suite (TTINF file must exist)

- Initialize a validation
  Call this command for the current validation, then for the reference validation. Both validations must be associated to the same test suite, which must be already initialized.

- Call the "Operations on sets" menu
  Use set operations to create sets of tests to be submitted to the "Analyze" command, and to the "Compare" command.

  The predefined sets ANALYZE_SET and COMPARE_SET may be used as initial sets, that you may modify to obtain the exact sets that you need.

- Analyze execution result files
  Use ANALYZE_SET or a derived set.

- Compare compilation/link result files
  Use COMPARE_SET or a derived set.

- Extract subsets from a set
  Construct the set of the tests having the status TO_BE_CHECKED_BY_HAND. Save this set into a file. Do the same thing with the status UNKNOWN_YET, PASSED_AND_SPECIAL, INAPPL_AND_SPECIAL.

- Display test information
  Use this command to create files containing the names of the test of the sets just created (for the TO_BE_CHECKED_BY_HAND tests, ...).

- Quit the program to check the logbook file of the current validation
  See what happened to the TO_BE_CHECKED_BY_HAND tests, and to the NONCONFORMANT tests. The logbook file also contains the full text of the execution result files of the SPECIAL tests: this is convenient for filling the validation logbook. The logbook file may be printed, the special actions required for the special tests may be done, then the tool may be launched again:

- Set a validation as the current validation
  To load the previous status file and all the validation information.

- Read a set from file
  Load the previously saved sets:
  - PASSED_AND_SPECIAL tests
  - INAPPL_AND_SPECIAL tests
  - etc.

- Update test status
  For each test of these 2 sets, update the status (to PASSED, INAPPLICABLE, or ...), according to the issue of the special actions previously done. Update also the status of the tests that have been checked by hand.

- At this point, only a few tests should remain, that do not have the status PASSED, INAPPLICABLE, or FAILED.
  These test must probably be treated one by one.